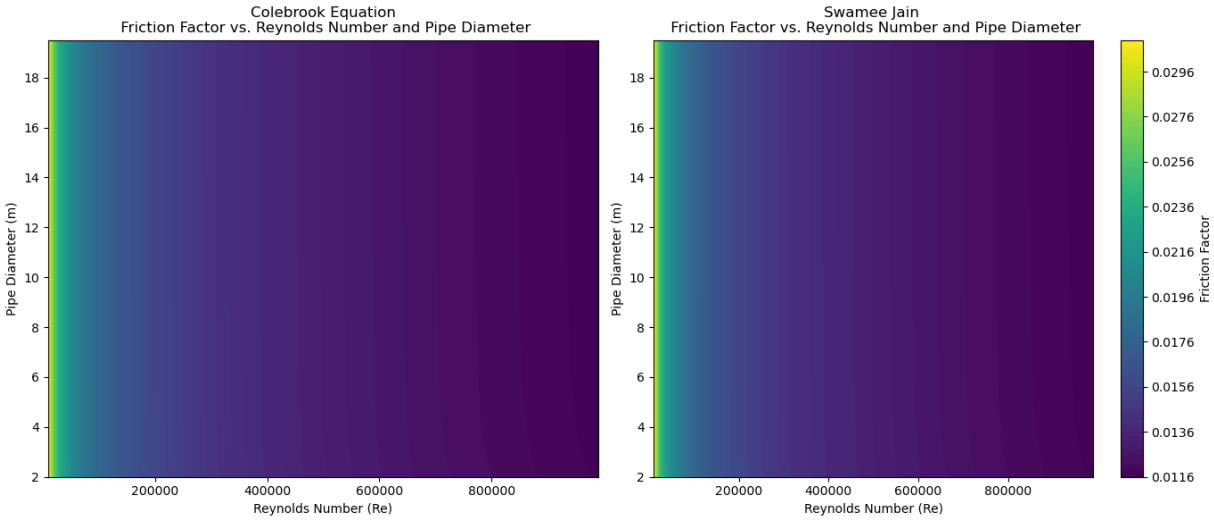
J Barton
A02298690
MAE2450-002 Spring 2025

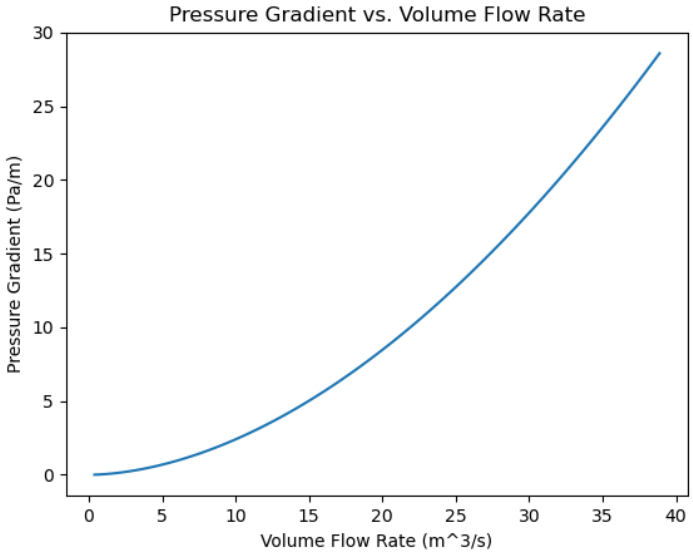
PROGRAMING HOMEWORK 2

PROBLEM 1

Graph generated from section A and B of problem 1, error is at 0.0001 and will do up to 1000 iterations.



Part C generates a pressure gradient graph as shown below. I used the Swamee Jian calculation because it was less computationally intense to allow for quicker computation.



J Barton
A02298690
MAE2450-002 Spring 2025

PROBLEM 2

A) Gives user the option to define their own NxN matrix or generate a random one
B) Comparing 9x9 matrices, below is shown 3 outputs

```
What is the size of the N x N matrix? 9
Do you want to enter a matrix(e) or create a random one(r)? (Enter 'e' or 'r') r
The System to be solved is:

Augmented Matrix:
{A |b}
[[0.18905952 0.15284385 0.16118212 0.9124678  0.82399624 0.64706315
  0.07172312 0.97195448 0.43029537 0.26756318]
 [0.63094362 0.84217533 0.33598134 0.85713436 0.19091661 0.32815473
  0.64060426 0.45647264 0.11468745 0.53807092]
 [0.08003261 0.65089896 0.24558573 0.09528044 0.78239025 0.7993223
  0.1245554  0.99094164 0.57666313 0.39167317]
 [0.78230278 0.8842666  0.34423639 0.44149186 0.91429637 0.99549912
  0.66549453 0.72784157 0.78499597 0.59850753]
 [0.72095109 0.99831625 0.64528943 0.26787604 0.5893282  0.36173056
  0.29960736 0.34745543 0.10499318 0.30283435]
 [0.37685872 0.42810763 0.78657713 0.84361731 0.58686839 0.67822049
  0.64837289 0.42980613 0.68955048 0.16221765]
 [0.86185514 0.40529753 0.33217663 0.10116407 0.73058581 0.12573104
  0.38573559 0.64150954 0.57060696 0.6154362 ]
 [0.68379893 0.26557219 0.37664716 0.33123434 0.30097686 0.38183621
  0.35863452 0.60478744 0.46339421 0.3433111 ]
 [0.93341546 0.0761893  0.62277852 0.7745385  0.76281634 0.1631467
  0.8053753  0.27867226 0.56071299 0.17730297]]

The Solution Matrix is:
Row 1: 0.2072
Row 2: 0.7465
Row 3: -0.5804
Row 4: 0.2700
Row 5: -0.2196
Row 6: -0.6222
Row 7: -0.3002
Row 8: 0.1499
Row 9: 0.9779
The python library to solve matrix is np.linalg.solve running this with the matrix results in:
[ 0.20724169  0.74653608 -0.58039304  0.27001053 -0.21959862 -0.62220244
 -0.3002354   0.14993733  0.97794696]
```

J Barton
A02298690
MAE2450-002 Spring 2025

```
What is the size of the N x N matrix? 9
Do you want to enter a matrix(e) or create a random one(r)? (Enter 'e' or 'r') r
The System to be solved is:

Augmented Matrix:
{A |b}
[[0.73967921 0.7750089  0.88734186 0.2401425  0.98503406 0.0766925
  0.56523842 0.50961864 0.56259012 0.23526292]
 [0.68560891 0.4262135  0.54873566 0.40023863 0.85364412 0.73668271
  0.86790098 0.80426681 0.50903294 0.96158612]
 [0.15621978 0.34271739 0.3277699  0.81316835 0.43736525 0.01308562
  0.74534973 0.02131742 0.69930542 0.96418221]
 [0.76938957 0.90710527 0.28497929 0.48110352 0.92886713 0.04121548
  0.06689255 0.09297682 0.71611446 0.56598728]
 [0.15497808 0.97779821 0.05078293 0.43058912 0.5225979  0.43568819
  0.0168471  0.84212442 0.40876381 0.46445736]
 [0.92386955 0.77184023 0.17408286 0.48444963 0.76285173 0.85058248
  0.9773908  0.30907672 0.81974642 0.59985743]
 [0.76693926 0.18012582 0.42707794 0.40388006 0.97125952 0.14633459
  0.51245414 0.76688008 0.09167963 0.57548104]
 [0.84011277 0.3419622  0.88328517 0.43873206 0.20711254 0.19021137
  0.25468329 0.16460719 0.92309722 0.01557948]
 [0.55967692 0.66495801 0.07279551 0.81425659 0.22044549 0.06426064
  0.30441322 0.10149635 0.38464844 0.59875783]]

The Solution Matrix is:
Row 1: -1.1810
Row 2: -0.2333
Row 3: 1.4184
Row 4: 2.2820
Row 5: 1.5198
Row 6: 2.3522
Row 7: -0.8898
Row 8: -1.5982
Row 9: -1.5589
The python library to solve matrix is np.linalg.solve running this with the matrix results in:
[-1.18098453 -0.23333243  1.41839654  2.28200797  1.51980384  2.35218622
 -0.88978663 -1.59817249 -1.55888963]
```

J Barton
A02298690
MAE2450-002 Spring 2025

```
What is the size of the N x N matrix? 9
Do you want to enter a matrix(e) or create a random one(r)? (Enter 'e' or 'r') r
The System to be solved is:

Augmented Matrix:
{A |b}
[[0.06297068 0.51351954 0.47033541 0.97247983 0.69826357 0.67522991
  0.8718404  0.38866243 0.13822526 0.32458965]
 [0.53294798 0.51789368 0.41787482 0.99515269 0.33156104 0.4524965
  0.14083501 0.20874775 0.02682289 0.66472982]
 [0.93425897 0.19616218 0.07918534 0.74830199 0.24164035 0.26073181
  0.66019125 0.91241365 0.62770419 0.93928857]
 [0.25044207 0.79232666 0.3952907  0.6023349  0.9928095  0.417347
  0.27495605 0.89056201 0.86376926 0.16042381]
 [0.08690449 0.09870477 0.51135071 0.50627182 0.86896717 0.67321183
  0.24770402 0.66164731 0.5105528  0.28060927]
 [0.47942014 0.00582081 0.63789461 0.6676935  0.79789539 0.82209994
  0.18932832 0.60241913 0.97503545 0.20895426]
 [0.50365947 0.04389189 0.57220009 0.24813688 0.1676944  0.29571995
  0.60202307 0.75779176 0.96425626 0.96762262]
 [0.12851047 0.85093905 0.44652011 0.18860134 0.84723475 0.97128067
  0.63691507 0.32171587 0.03750602 0.08005206]
 [0.49137147 0.70560098 0.77051827 0.20914692 0.94022227 0.17264197
  0.3300174  0.95247224 0.47515374 0.74602331]]

The Solution Matrix is:
Row 1: -0.0038
Row 2: 0.1562
Row 3: 0.9749
Row 4: 0.2124
Row 5: -1.4715
Row 6: 0.3669
Row 7: -0.2030
Row 8: 1.6016
Row 9: -0.6233
The python library to solve matrix is np.linalg.solve running this with the matrix results in:
[-0.00383008  0.15623183  0.97486729  0.21235836 -1.47150258  0.36694569
 -0.20303942  1.60156027 -0.6232616 ]
```

J Barton
A02298690
MAE2450-002 Spring 2025

PROBLEM 3

A) The code generates the LU decomposition for the Crout version

```python
def crout(A):
    L = zero_matrix(n)
    U = identity_matrix(n)

    for i in range(n):
        #Lower Triangular Matrix L
        for j in range(i, n):
            sum_val = sum(L[j][k] * U[k][i] for k in range(i))
            L[j][i] = A[j][i] - sum_val


        #Upper Triangle
        for j in range(i+1, n):
            sum_val = sum(L[i][k] * U[k][j] for k in range(i))
            if np.isclose(L[i][i], 0, atol=1e-10):
                raise ValueError("Matrix is singular. No inverse exists.")
            U[i][j] = (A[i][j] - sum_val)/ L[i][i]
    return L, U

def forward_substitution(L, b):
    n = len(b)
    y = np.zeros(n)
    for i in range(n):
        y[i] = b[i]
        for j in range(i):
            y[i] -= L[i][j] * y[j]
        y[i] = y[i] / L[i][i]
    return y

def backward_substitution(U, y):
    n = len(y)
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = y[i]
        for j in range(i+1, n):
            x[i] -= U[i][j] * x[j]
        x[i] = x[i] / U[i][i]
    return x
```

B) Shows 3 implementations or the code, one thing that I noticed while computing the identity matrix with the generated inverse I noticed that I was getting very small numbers, numbers with an order of magnitude of 10^-16 this is due to python carrying many significant figures that most of the time aren't needed so this anything at this magnitude is essential zero, so I use np.abs(np.round()) to generate the correct looking identity matrix

J Barton
A02298690
MAE2450-002 Spring 2025

```
What is the size of the N x N matrix? 3
Do you want to enter the matrix manually? (y/n): n

Matrix:
[[ 2 -1  0]
 [ 8  7 -3]
 [ 3  8 -8]]

Lower Triangular Matrix:
[[ 2.          0.          0.        ]
 [ 8.         11.          0.        ]
 [ 3.          9.5        -5.40909091]]

Upper Triangular Matrix:
[[ 1.         -0.5         0.        ]
 [ 0.          1.         -0.27272727]
 [ 0.          0.          1.        ]]

Inverse Matrix:
[[ 0.26890756  0.06722689 -0.02521008]
 [-0.46218487  0.13445378 -0.05042017]
 [-0.36134454  0.15966387 -0.18487395]]
Python function Inversere:
[[ 0.26890756  0.06722689 -0.02521008]
 [-0.46218487  0.13445378 -0.05042017]
 [-0.36134454  0.15966387 -0.18487395]]

Verification:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

```
What is the size of the N x N matrix? 3
Do you want to enter the matrix manually? (y/n): n

Matrix:
[[-5  3 -9]
 [-7  6 -8]
 [ 3  7 -7]]

Lower Triangular Matrix:
[[ -5.           0.           0.        ]
 [ -7.           1.8          0.        ]
 [  3.           8.8        -34.88888889]]

Upper Triangular Matrix:
[[ 1.         -0.6         1.8       ]
 [ 0.          1.          2.55555556]
 [ 0.          0.          1.        ]]

Inverse Matrix:
[[ 0.04458599 -0.13375796  0.0955414 ]
 [-0.23248408  0.19745223  0.07324841]
 [-0.2133758   0.14012739 -0.02866242]]
Python function Inversere:
[[ 0.04458599 -0.13375796  0.0955414 ]
 [-0.23248408  0.19745223  0.07324841]
 [-0.2133758   0.14012739 -0.02866242]]

Verification:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

```
What is the size of the N x N matrix? 3
Do you want to enter the matrix manually? (y/n): n

Matrix:
[[ 5 -7 -9]
 [ 5 -3  7]
 [-7  9  6]]

Lower Triangular Matrix:
[[ 5.   0.   0. ]
 [ 5.   4.   0. ]
 [-7.  -0.8 -3.4]]

Upper Triangular Matrix:
[[ 1.  -1.4 -1.8]
 [ 0.   1.   4. ]
 [ 0.   0.   1. ]]

Inverse Matrix:
[[ 1.19117647  0.57352941  1.11764706]
 [ 1.16176471  0.48529412  1.17647059]
 [-0.35294118 -0.05882353 -0.29411765]]
Python function Inversere:
[[ 1.19117647  0.57352941  1.11764706]
 [ 1.16176471  0.48529412  1.17647059]
 [-0.35294118 -0.05882353 -0.29411765]]

Verification:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```